computer then uses the classification of words 100A to label each chunk with the class, sub-class and word category of the core word.

Each non-chunk word is similarly labelled on the basis of the classification of words 100A, as is each piece of punctuation.

The classifications 101 for the elements generated by the classification process 100 are stored in RAM 12.

Returning again to the example sentence, after classification of the elements of the input sentence would be as shown in Table 5 below

CLASS = [sentstart]
phrasetag(<ADV) CLASS = [adv] Similarly_RR
CLASS = [punct minpunct] ,_,
phrasetag((NR) CLASS = [nonreferent proper] Britain_NP1
phrasetag([VG) CLASS = [vg past] became_VVD
phrasetag(<ADJ) CLASS = [adj] popular_JJ
phrasetag([pp) CLASS = [pp icspp after] after_ICS
phrasetag ([pp) CLASS = [pp icspp after] after_ICS << SUBCAT phrasetag((NR) CLASS = [nonreferent] a_AT1 rumour_NN1 >>
phrasetag([VG) CLASS = [vg verbpart] got_VVD about_RP
CLASS = I [lex coords cst] that_CST
phrasetag((NR) CLASS = [nonreferent proper place titular] Mrs_NNSB1 Thatcher_NP1
phrasetag([VG) CLASS = [vg past] had_VHD declared_VVN
phrasetag(NR CLASS = [nonreferent locative] open_JJ house_NNL1 NR)
CLASS = [punct majpunct] ._.
CLASS = [sentend]

Table 5

It will be seen that each element is labelled with a class and also a sub-class where there are a number of word categories within the sub-class.

Returning to Figure 2A, as stated above, the syntactic information 48 and word grouping data 46 are stored in the RAM 12 by the text analysis process 42. The syntactic information 48 comprises word tags 95, syntactic groups 96, chunk markers 99 and element classifications 101. The word grouping data comprises the sentence markers 86 and paragraph markers 87.

Similar processing is carried out in forming the prosodic structure corpus 52 stored on the CD-ROM 32. Therefore, each of the reference sentences within the corpus is divided into elements and has similar syntactic information relating to each of the elements contained within it. Furthermore, the corpus contains data indicating where a human would insert prosodic boundaries when reading each of the example sentences. The type of the boundary is also indicated.

An example of the beginning of a sentence that might be found in the corpus 52 is given in Table 6 below. In Table 6, the absence of a boundary is shown by the label 'sfNONE' after an element, the presence of a boundary is shown by 'sfMINOR' or 'sfMAJOR' depending on the strength of the boundary. The start of the example sentence is "As ever , | the American public | and the world 's press | are hungry for drama..."

CLASS = [sentstart] sfNONE
phrasetag(<ADV) CLASS = [adv] As_RG ever_RR sfNONE
CLASS = [punct minpunct] ,_, sfMINOR
phrasetag((NR) CLASS = [nonreferent] the_AT American_JJ public_NN sfMINOR
CLASS = [lex coords cc] and_CC sfNONE
phrasetag((NR) CLASS = [nonreferent] the_AT world_NN1 's_$ press_NN sfMINOR
phrasetag([VG) CLASS = [vg beverbs] are_VBR sfNONE
phrasetag(<ADJ) CLASS = [adj] hungry_JJ sfNONE

phrasetag([pp) CLASS = [pp ifpp for] for_IF << SUBCAT phrase tag((NR) CLASS = [nonreferent] drama_NN1 sfNONE >>

Table 6

The prosodic structure prediction process 50 involves the computer in finding the sequence of elements in the corpus which best matches a search sequence taken from the input sentence. The degree of matching is found in terms of syntactic characteristics of corresponding elements, length of the elements in words and a comparison of boundaries in the reference sentence and those already predicted for the input sentence. The process 50 will now be described in more detail with reference to Figure 5.

Figure 5 shows that the process 50 begins with the calculation of measures of similarity between each element of the input sentence and each element of the corpus 52. This part of the program is presented in the form of pseudo-code below:

```
FOR each element(e_i) of the input sentence:
        FOR each element(e_r) of the corpus:
                calculate degree of syntactic match between elements e_i and e_r (=A)
                calculate no._of_words match between elements e_i and e_r (=B)
                calculate  syntactic match between words in elements e_i and e_r (=C)
                match(e_i,e_r) = w1*A + w2 * B + w3 * C
        NEXT e_r
NEXT e_i
```

where e_i increments from 1 to the number of elements in the input sentence, and e_r increments from 1 to the number of elements in the corpus.

In order to calculate the degree of syntactic match between elements, the program controls the computer to find: